

## Цель работы

Написать функцию, которая возвращает кортеж из двух индексов элементов списка `lst`, таких что сумма элементов по этим индексам равна переменной `target`.

Элемент по индексу может быть выбран лишь единожды, значения в списке могут повторяться.

Если в списке встречается больше чем два индекса, подходящих под условие - вернуть наименьшие из всех. Элементы находятся в списке в произвольном порядке.

## Комментарии по выполнению

### Стартовый борд в repl.it

Для входного набора

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
target = 8
```

программа должна вернуть

```
(0, 6)
```

Для начала решим эту задачу "в лоб" с помощью циклов

```
def two_sum(lst, target):
    pass
```

## Усложнение 1

Решим задачу другим способом, чтобы вычислительная сложность решения была не  $O(n^2)$ , а ниже (например,  $O(n)$ ).

Решение оформить в виде отдельной функции:

```
def two_sum_hashed(lst, target):
    pass
```

## Усложнение 2

Усложним задачу и вернем все наборы индексов, удовлетворяющих условию.

Для входного набора:

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
target = 8
```

программа должна вернуть:

```
[(0,6), (1,5), (2,4)]
```

Опубликуйте решение с помощью сервиса [repl.it](#) и предоставьте в конце занятия ссылку на собственное решение в [repl.it](#)

---

```
def main():
    lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
    target = 8
    print("First result is:")
    two_sum(lst, target)
    two_sum_hashed(lst, target)

def two_sum(lst, target):
    for i in range(len(lst) - 1):
        for j in range(i + 1, len(lst)):
            if lst[j] == target - lst[i]:
                print("({0},{1})".format(i, j))

def two_sum_hashed(lst, target):
    listF = {}
    for k in range(len(lst)):
        listF[k] = lst[k]
    print("\nDictionary of original list:")
    print("{0}".format(listF))
    newList = {}
    for k in range(len(lst)):
        newList[k] = target - listF[k]
    print("\nDictionary of difference between values and target:")
    print("{0}".format(newList))
    slovar = {}
    for k in range(len(lst)):
        keys = list(listF.keys())
        values = list(listF.values())
        num = newList.get(k)
        if num in values[k + 1:len(lst)]:
            num = values.index(num, k + 1, len(lst))
            keyValue = keys.pop(num)
            slovar.setdefault(k, [keyValue])
    print("\nItems of result dictionary:")
    print("{0}".format(slovar.items()))

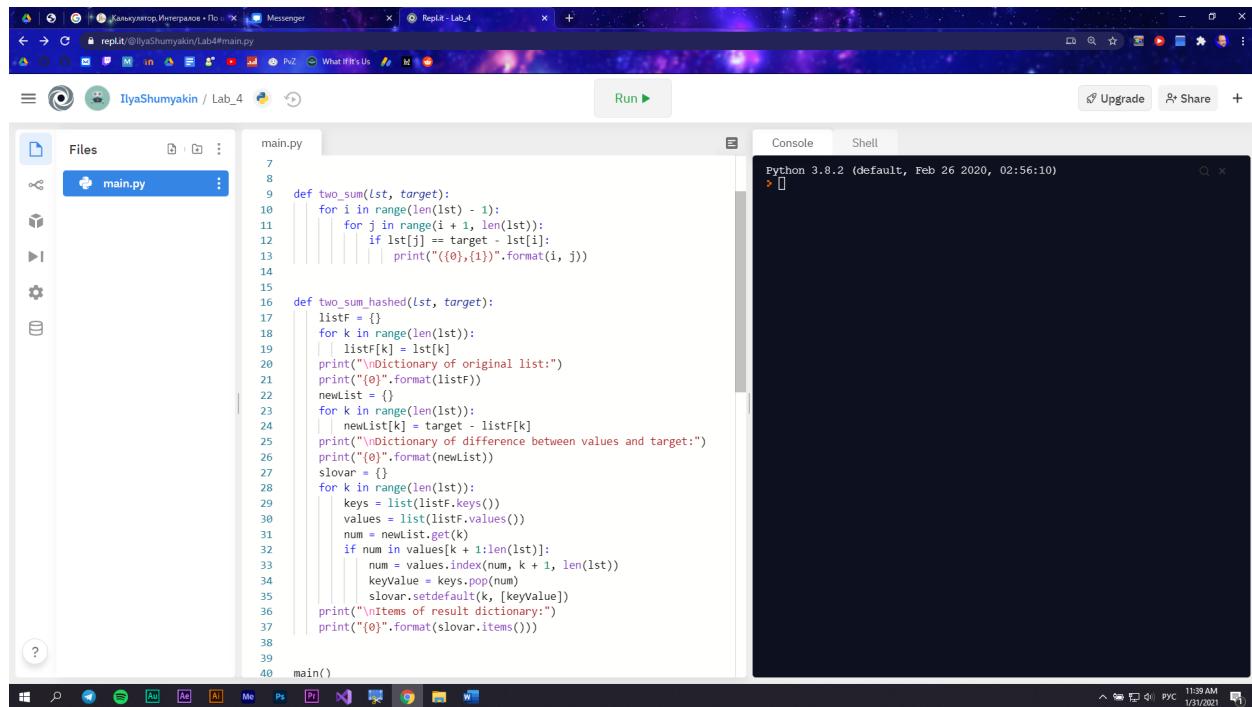
main()
# + 1 формирование словаря из листа
# + 2 вычисление для каждого элемента листа его разницы diff с target
# + 3 поиск в словаре значения разницы
# + 4 проверить не совпадают ли индексы двух найденных элементов на шаге 2 и 3
```

```
# Подсказка: для шага 4 хорошо использовать методы keys или values или items, get у объекта типа dict.
```

```
# Усложнение 1 Решим задачу другим способом, чтобы вычислительная сложность решения была не O(n^2), а ниже (например, O(n)).
```

```
# Усложнение 2 Усложним задачу и вернем все наборы индексов, удовлетворяющих условию. Программа должна вернуть: [(0,6), (1,5), (2,4)]
```

<https://repl.it/@IlyaShumyakin/Lab4#main.py>



The screenshot shows a Repl.it interface. On the left is a code editor with a 'Files' sidebar containing a single file 'main.py'. The code in 'main.py' is as follows:

```
7
8
9  def two_sum(List, target):
10     for i in range(len(List) - 1):
11         for j in range(i + 1, len(List)):
12             if List[j] == target - List[i]:
13                 print("({0},{1})".format(i, j))
14
15
16  def two_sum_hashed(List, target):
17      listF = {}
18      for k in range(len(List)):
19          listF[k] = List[k]
20      print("\nDictionary of original list:")
21      print("{0}".format(listF))
22      newList = {}
23      for k in range(len(List)):
24          newList[k] = target - listF[k]
25      print("\nDictionary of difference between values and target:")
26      print("{0}".format(newList))
27      slovar = {}
28      for k in range(len(List)):
29          keys = list(listF.keys())
30          values = list(listF.values())
31          num = newList.get(k)
32          if num in values[k + 1:len(List)]:
33              num = values.index(num, k + 1, len(List))
34              keyValue = keys.pop(num)
35              slovar.setdefault(k, [keyValue])
36      print("\nItems of result dictionary:")
37      print("{0}".format(slovar.items()))
38
39
40  main()
```

On the right is a terminal window titled 'Console' showing the output of the Python 3.8.2 interpreter. The output is:

```
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
[1]:
```

The screenshot shows a Python code editor interface with a file named 'main.py' open. The code implements a function to find pairs of elements in a list whose difference is a target value. The code uses a dictionary to store the list elements and then iterates through the list to find pairs. The console window shows the Python interpreter running and awaiting input.

```
main.py
28 for k in range(len(lst)):
29     keys = list(lst.keys())
30     values = list(lst.values())
31     num = newList.get(k)
32     if num in values[k + 1:len(lst)]:
33         num = values.index(num, k + 1, len(lst))
34         keyValue = keys.pop(num)
35         slovar.setdefault(k, [keyValue])
36         print("\nItems of result dictionary:")
37         print("{0}".format(slovar.items()))
38
39 def main():
40     # + 1 формирование словаря из листа
41     # + 2 вычисление для каждого элемента листа его разницы diff с target
42     # + 3 поиск в словаре значения разницы
43     # + 4 проверить не совпадают ли индексы двух найденных элементов на
44     # шаге 2 и 3
45
46     # Подсказка: для шага 4 хорошо использовать методы keys или values
47     # или items, get у объекта типа dict.
48
49     # Усложнение 1 Решим задачу другим способом, чтобы вычислительная
50     # сложность решения была не O(n^2), а ниже (например, O(n)).
```

Console

```
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
> []
```